## APPENDIX A

| Class Diagram Summary | |
|---|---|
| CableMoose_DA | Contains all of the classes needed to build an OpenCable compliant CableMoose Distributed Application. |

| Class Summary | |
|---|---|
| AppCmdStates | A DA network partition (DA_NetPart) can send a message (a.k.a. |
| DA_Cmd | Application Commands can be instigated by either a DASP or DACP. |
| DA_Cmd.TimerListener | |
| DA_IPCPart | Inter-Process Communications (IPC) are intended for Peer Partitions on the same machine. |
| DA_Msg | Implements a CableMoose message. |
| DA_NetPart | The DA Network Partition implements the CableMoose proprietary network communications protocol. |
| DA_Partition | Base of the CableMoose communications hierarchy. |
| DA_Portion | |
| DA_Session | |
| DACP_Framework | Base of the OpenCable compliant DACP Framework hierarchy. |
| DASP_Framework | Base of the load balancing DA Server portion inheritance hierarchy. |
| MooseCmdTypes | |
| MooseFault | |
| ProximityDetector | Stand-alone object that can determine the proximity of one DA Virtual Address to another. |

CableMoose_DA

# Class AppCmdStates

public class **AppCmdStates**

A DA network partition (DA_NetPart) can send a message (a.k.a. command) as a client or receive a message as a server. If it's sending it can wait for the message synchronously or asynchronously. Each command object will store its state as set by the DA_NetPart. A Client state machine will start in the INIT_S state. When sent, the command will move to the CLIENT_ACK_WAIT state. When an ACK is received it will move to the DATA_WAIT state if the command requires a data response. If not, the command will move to the DONE_S state and be discarded. A Server state machine will start in the SERVER_RECVD_MSG state when a command is received. It will ACK the msg if valid and move to the ANS_MSG state if the command requires a data response. If not, the command will move to the DONE_S and be discarded. If the command does require a data response, the server processes the command, creates the data, sends it to the client and moves the command to the DONE_S where it is discarded.

## Field Summary

| | |
|---|---|
| static int | CLIENT_ACK_WAIT_S<br>Client wait for acknowledge state. |
| static int | CLIENT_DATA_WAIT_S<br>Client wait for data state. |
| static int | DONE_S<br>Command done state. |
| static int | INIT_S<br>Initialization state. |
| static int | SERVER_ACK_MSG_S<br>Server acknowledge message state. |
| static int | SERVER_ANS_MSG_S<br>Server answer message state. |
| static int | SERVER_RECVD_MSG_S<br>Server message received state. |

## Field Detail

### CLIENT_ACK_WAIT_S

```
public static final int CLIENT_ACK_WAIT_S
```

Client wait for acknowledge state.

---

## CLIENT_DATA_WAIT_S

```
public static final int CLIENT_DATA_WAIT_S
```

Client wait for data state.

---

## DONE_S

```
public static final int DONE_S
```

Command done state.

---

## INIT_S

```
public static final int INIT_S
```

Initialization state.

---

## SERVER_ACK_MSG_S

```
public static final int SERVER_ACK_MSG_S
```

Server acknowledge message state.

---

## SERVER_ANS_MSG_S

```
public static final int SERVER_ANS_MSG_S
```

Server answer message state.

---

## SERVER_RECVD_MSG_S

```
public static final int SERVER_RECVD_MSG_S
```

Server message received state.

---

CableMoose_DA

# Class DA_Cmd

public class **DA_Cmd**
extends Object

Application Commands can be instigated by either a DASP or DACP. Each command is stand-alone containing state, destination address, and message buffers.

## Inner Class Summary

| | |
|---|---|
| private class | DA_Cmd.TimerListener |

## Field Summary

| | |
|---|---|
| private int | ackTimeout |
| private boolean | answerExpected |
| private int | dataTimeout |
| private String | destVA |
| byte | MAX_RETRIES |
| private DA_Msg | msg |
| private byte | retries |
| private int | state |
| private Timer | timer |

## Constructor Summary

| |
|---|
| DA_Cmd(DA_Msg newMsg) |
| DA_Cmd(int newDataTimeout, int newAckTimeout, DA_Msg newMsg) |

## Method Summary

| | |
|---|---|
| int | getAckTimeout() |
| boolean | getAnsExpected() |
| DA_Msg | getDaMsg() |
| int | getDataTimeout() |
| int | getState() |
| boolean | incrAndTestRetries() |
| void | setAckTimeout(int newAckTimeout) |
| void | setDataTimeout(int newDataTimeout) |
| void | setState(int newState) |
| void | startTimeout(int timeout) |

## Field Detail

### ackTimeout

```
private int ackTimeout
```

---

### answerExpected

```
private boolean answerExpected
```

---

### dataTimeout

```
private int dataTimeout
```

### destVA

```
private String destVA
```

---

### MAX_RETRIES

```
public final byte MAX_RETRIES
```

---

### msg

```
private DA_Msg msg
```

link aggregationByValue
associates
supplierRole Msg
supplierCardinality 1

---

### retries

```
private byte retries
```

---

### state

```
private int state
```

---

### timer

```
private Timer timer
```

---

## Constructor Detail

### DA_Cmd

```
public DA_Cmd(DA_Msg newMsg)
```

---

### DA_Cmd

public DA_Cmd(int newDataTimeout, int newAckTimeout, DA_Msg newMsg)

# Method Detail

## getAckTimeout

public int getAckTimeout()

---

## getAnsExpected

public boolean getAnsExpected()

---

## getDaMsg

public DA_Msg getDaMsg()

---

## getDataTimeout

public int getDataTimeout()

---

## getState

public int getState()

---

## incrAndTestRetries

public boolean incrAndTestRetries()

---

## setAckTimeout

public void setAckTimeout(int newAckTimeout)

---

## setDataTimeout

public void setDataTimeout(int newDataTimeout)

---

## setState

```
public void setState(int newState)
```

---

### startTimeout

```
public void startTimeout(int timeout)
```

## Association Links

to Class CableMoose_DA.DA_Msg

Attribute msg
Supplier Role Msg
Supplier Cardinality 1
Type Composition

to Class javax.swing.Timer

Attribute timer

CableMoose_DA

# Class DA_Cmd.TimerListener

private class DA_Cmd.TimerListener
implements ActionListener

## Method Summary

| | |
|---|---|
| void | actionPerformed(ActionEvent e) |

## Method Detail

### actionPerformed

```
public void actionPerformed(ActionEvent e)
```

CableMoose_DA

# Class DA_IPCPart

```
CableMoose_DA.DA_Portion
   |
   +--CableMoose_DA.DA_Partition
         |
         +--CableMoose_DA.DA_IPCPart
```

---

public class DA_IPCPart
extends DA_Partition
implements Runnable

Inter-Process Communications (IPC) are intended for Peer Partitions on the same machine. IPC is not necessary for mutable Client/Server Partitions and can't be used for inter-machine comm.

---

## Field Summary

| | |
|---|---|
| private Vector | AppCmds |

Fields inherited from class CableMoose_DA.DA_Portion

clientProximityDetector, clientServerParts, peerParts

## Constructor Summary

DA_IPCPart()

## Method Summary

| | |
|---|---|
| void | ackMsg(byte [] data) |
| void | destMoveNotify(String oldDestVA, String newDestVA) |
| void | nakMsg(byte [] data, short reason) |
| DA_Cmd | recvMsg() |
| void | run() |
| void | sendMsg(DA_Cmd cmd) |

Methods inherited from class CableMoose_DA.DA_Partition

cmdReply, getNextCmd

## Field Detail

### AppCmds

```
private Vector AppCmds
```

link aggregationByValue
associates
supplierCardinality 0..*

## Constructor Detail

### DA_IPCPart

```
public DA_IPCPart()
```

## Method Detail

### ackMsg

```
public void ackMsg(byte [] data)
```

---

### destMoveNotify

```
public void destMoveNotify(String oldDestVA, String newDestVA)
```

---

### nakMsg

```
public void nakMsg(byte [] data, short reason)
```

---

### recvMsg

```
public DA_Cmd recvMsg()
```

---

### run

```
public void run()
```

---

### sendMsg

```
public void sendMsg(DA_Cmd cmd)
```

## Association Links

to Class CableMoose_DA.DA_Session

Attribute AppCmds
Supplier Cardinality 0..*
Type Composition

CableMoose_DA

# Class DA_Msg

---

public class **DA_Msg**

Implements a CableMoose message. Because UDP messages have an effective maximum size of 512 bytes, a CableMoose message may span multiple data buffers. In addtion, CableMoose adds a proprietary header of 8 or 12 bytes reducing the effective message size even further. The CableMoose specific header will have the following format: cmdType : short // Command Type, 0 through 0xFF are reserved for CableMoose sessionId : byte // Session Id, each DASP can have 255 open sessions. xactId : byte // Transaction Id, unique number derived by message sender.. flags : byte // b7 indicates number of UDP buffers > 1. // b6 indicates message is an ACK. // b5 indicates message is a NAK. sentTime : 3 bytes // Low order 10 bits are milliseconds, // next 6 bits are minutes, // next 6 bits are hours. If b7 is set the message will contain more than 1 UDP data buffer and the following header is included as well: bufNum : short // Buffer number of this message, used for sequencing. totalNumBufs : short // Total number of buffers in the message Data past the header is application specific and must be implemented by deriving classes.

---

## Field Summary

| | |
|---|---|
| static byte | ACK_MASK |
| static byte | BASIC_HDR_LEN |
| static byte | BIG_BUF_HDR_SIZE |
| static byte | BUF_NUM_IDX |
| private Bits | bufBitSet |
| private byte[] | buffer |
| private int | bufIdx |
| static byte | BUFS_MASK |
| static byte | CMD_TYPE_IDX |
| private byte | cmdType |

| | |
|---|---|
| private int | curBufNum |
| private int | dataLen |
| private byte | flags |
| static byte | FLAGS_IDX |
| static byte | HDR_SIZE |
| static byte | LAST_BUF_LEN_IDX |
| static short | MAX_BUF_LEN |
| static short | MAX_DATA_LEN |
| static byte | MULTI_BUF_HDR_LEN |
| static byte | NAK_MASK |
| private short | numBufs |
| static byte | SENT_TIME_IDX |
| private int | sentTime |
| static byte | SESS_ID_IDX |
| private byte | sessionId |
| static byte | TOT_BUFS_IDX |
| private short | totNumBufs |
| static byte | XACT_ID_IDX |
| private byte | xactId |

## Constructor Summary

| |
|---|
| DA_Msg(byte [] newMsg) |
| DA_Msg() |

## Method Summary

| | |
|---|---|
| void | addData(byte [] newMsg) |
| boolean | allBufsRecvd() |
| byte | getCmdType() |
| byte [] | getData() |
| byte | getFlags() |
| int | getSentTime() |
| byte | getSessionId() |
| byte | getXactId() |
| static byte | parseBufNum(byte [] msg) |
| static byte | parseCmdType(byte [] msg) |
| static byte | parseFlags(byte [] msg) |
| static boolean | parseMultiBufsFlag(byte [] msg) |
| static byte | parseSession(byte [] msg) |
| static byte | parseXactId(byte [] msg) |
| void | setbuffer() |
| void | setCmdType(byte newCmdType) |
| void | setFlags(byte newFlags) |
| void | setSentTime(int newSentTime) |
| void | setSessionId(byte newSessionId) |

| | |
|---|---|
| void | setXactId(byte newXactId) |

# Field Detail

## ACK_MASK

```
public static final byte ACK_MASK
```

---

## BASIC_HDR_LEN

```
public static final byte BASIC_HDR_LEN
```

---

## BIG_BUF_HDR_SIZE

```
public static byte BIG_BUF_HDR_SIZE
```

---

## BUF_NUM_IDX

```
public static final byte BUF_NUM_IDX
```

---

## bufBitSet

```
private Bits bufBitSet
```

---

## buffer

```
private byte[] buffer
```

---

## bufIdx

```
private int bufIdx
```

---

## BUFS_MASK

```
public static final byte BUFS_MASK
```

## CMD_TYPE_IDX

public static final byte CMD_TYPE_IDX

---

## cmdType

private byte cmdType

---

## curBufNum

private int curBufNum

---

## dataLen

private int dataLen

---

## flags

private byte flags

---

## FLAGS_IDX

public static final byte FLAGS_IDX

---

## HDR_SIZE

public static final byte HDR_SIZE

---

## LAST_BUF_LEN_IDX

public static final byte LAST_BUF_LEN_IDX

---

## MAX_BUF_LEN

public static final short MAX_BUF_LEN

## MAX_DATA_LEN

```
public static final short MAX_DATA_LEN
```

---

## MULTI_BUF_HDR_LEN

```
public static final byte MULTI_BUF_HDR_LEN
```

---

## NAK_MASK

```
public static final byte NAK_MASK
```

---

## numBufs

```
private short numBufs
```

---

## SENT_TIME_IDX

```
public static final byte SENT_TIME_IDX
```

---

## sentTime

```
private int sentTime
```

---

## SESS_ID_IDX

```
public static final byte SESS_ID_IDX
```

---

## sessionId

```
private byte sessionId
```

---

## TOT_BUFS_IDX

```
public static final byte TOT_BUFS_IDX
```

### totNumBufs

```
private short totNumBufs
```

---

### XACT_ID_IDX

```
public static final byte XACT_ID_IDX
```

---

### xactId

```
private byte xactId
```

## Constructor Detail

### DA_Msg

```
public DA_Msg(byte [] newMsg)
```

---

### DA_Msg

```
public DA_Msg()
```

## Method Detail

### addData

```
public void addData(byte [] newMsg)
```

---

### allBufsRecvd

```
public boolean allBufsRecvd()
```

---

### getCmdType

```
public byte getCmdType()
```

---

### getData

```
public byte [] getData()
```

---

## getFlags

```
public byte getFlags()
```

---

## getSentTime

```
public int getSentTime()
```

---

## getSessionId

```
public byte getSessionId()
```

---

## getXactId

```
public byte getXactId()
```

---

## parseBufNum

```
public static byte parseBufNum(byte [] msg)
```

---

## parseCmdType

```
public static byte parseCmdType(byte [] msg)
```

---

## parseFlags

```
public static byte parseFlags(byte [] msg)
```

---

## parseMultiBufsFlag

```
public static boolean parseMultiBufsFlag(byte [] msg)
```

---

## parseSession

```
public static byte parseSession(byte [] msg)
```

---

### parseXactId

```
public static byte parseXactId(byte [] msg)
```

---

### setbuffer

```
public void setbuffer()
```

---

### setCmdType

```
public void setCmdType(byte newCmdType)
```

---

### setFlags

```
public void setFlags(byte newFlags)
```

---

### setSentTime

```
public void setSentTime(int newSentTime)
```

---

### setSessionId

```
public void setSessionId(byte newSessionId)
```

---

### setXactId

```
public void setXactId(byte newXactId)
```

## Association Links

to Class CableMoose_Utils.Bits

Attribute bufBitSet

---

CableMoose_DA

# Class DA_NetPart

```
CableMoose_DA.DA_Portion
  |
  +--CableMoose_DA.DA_Partition
        |
        +--CableMoose_DA.DA_NetPart
```

---

public class **DA_NetPart**
extends DA_Partition
implements Runnable

The DA Network Partition implements the CableMoose proprietary network communications protocol. It enables communications with multiple clients simultaneously without spawning a comm thread for each client. This is achieved with connectionless communications via the UDP protocol. Each command received will be checked for corruption and validity, then acked or naked accordingly.

---

## Field Summary

| | |
|---|---|
| private byte [] | ackNakBuff |
| private HashMap | appSessions |
| private byte [] | buff |
| private boolean | clientOnly |
| private byte | clientSessionId |
| private byte | curXactId |
| private Thread | daPortion |
| private InetAddress | destAddr |
| private int | nxtSessionIdNum |
| private LinkedList | readyCmds |
| private int | recvPort |

| | |
|---|---|
| private int | sendPort |
| private String | sourceVA |
| private DatagramPacket | udpPack |
| private DatagramSocket | udpSock |

| Fields inherited from class CableMoose_DA.DA_Portion |
|---|
| clientProximityDetector, clientServerParts, peerParts |

## Constructor Summary

| |
|---|
| DA_NetPart(InetAddress addr, int port, Thread daPortion, boolean client)<br>Primary constructor. |
| DA_NetPart(String destVA) |

## Method Summary

| | |
|---|---|
| void | ackMsg(byte [] data)<br>Verify cmd's destination command number, IP address and Port are valid. |
| void | closeSession()<br>Called by a client or server to close the session and remove the application from the server session list. |
| void | cmdReply() |
| void | destMoveNotify(String oldDestVA, String newDestVA) |
| DA_Cmd | getNextCmd()<br>Return the next command in the AppCmds structure. |
| String | getSourceVA() |
| void | handleAckNak(byte [] data) |
| void | handleAppMsg(DA_Session session, byte [] data)<br>Parse a message from an application. |
| void | handleCableMooseCmd(DA_Session session, byte [] data) |
| void | makePktHdr(byte [] msg, short cmdType, byte sessionId, byte xactId, byte flags) |
| void | nakMsg(byte [] data, short reason)<br>Verify cmd's destination command number, IP address and Port are valid. |

| | |
|---|---|
| void | nakMsg(InetAddress addr, int port, short reason) |
| void | openSession() |
| void | openSessionReply()<br>Acknowledge to an openSession message from a client. |
| DA_Cmd | recvMsg() |
| void | run()<br>Server thread for the network partition. |
| void | sendCmdMsg(AppSession session, DA_Cmd cmd) |
| void | sendMsg(AppSession session, DA_Cmd cmd) |
| void | setSourceVA(String newSourceVA) |

| Methods inherited from class CableMoose_DA.DA_Partition |
|---|
| cmdReply |

# Field Detail

## ackNakBuff

```
private byte [] ackNakBuff
```

---

## appSessions

```
private HashMap appSessions
```

link aggregationByValue
associates
directed
supplierCardinality 0..*

---

## buff

```
private byte [] buff
```

---

## clientOnly

```
private boolean clientOnly
```

## clientSessionId

```
private byte clientSessionId
```

---

## curXactId

```
private byte curXactId
```

---

## daPortion

```
private Thread daPortion
```

---

## destAddr

```
private InetAddress destAddr
```

---

## nxtSessionIdNum

```
private int nxtSessionIdNum
```

---

## readyCmds

```
private LinkedList readyCmds
```

---

## recvPort

```
private int recvPort
```

---

## sendPort

```
private int sendPort
```

---

## sourceVA

```
private String sourceVA
```

### udpPack

```
private DatagramPacket udpPack
```

---

### udpSock

```
private DatagramSocket udpSock
```

## Constructor Detail

### DA_NetPart

```
public DA_NetPart(InetAddress addr, int port, Thread daPortion, boolean client)
```

Primary constructor. If this is a client only partition a DatagramSocket won't be created and the server thread won't be started.

---

### DA_NetPart

```
public DA_NetPart(String destVA)
```

Preconditions - Formulation of a valid destination virtual address.
Semantics - Create a DA_NetPart Object with an empty AppCmds structure and a valid destination address.

## Method Detail

### ackMsg

```
public void ackMsg(byte [] data)
```

Verify cmd's destination command number, IP address and Port are valid. Send cmd Ack to sender.
Preconditions - Valid data parameter.
Postconditions - Acknowledge message sent to command sender.

---

### closeSession

```
public void closeSession()
```

Called by a client or server to close the session and remove the application from the server session list.

## cmdReply

```
public void cmdReply()
```

---

## destMoveNotify

```
public void destMoveNotify(String oldDestVA, String newDestVA)
```

Preconditions - Valid oldDestVA and newDestVA parameters. Must meet CableMoose virtual address format requirements.
Postconditions - All current AppCmd's with the oldDestVA are updated to the new.

---

## getNextCmd

```
public DA_Cmd getNextCmd()
```

Return the next command in the AppCmds structure. This allows a server Thread to handle the commands as it can. This method will call a synchronized method.

---

## getSourceVA

```
public String getSourceVA()
```

---

## handleAckNak

```
public void handleAckNak(byte [] data)
```

---

## handleAppMsg

```
public void handleAppMsg(DA_Session session, byte [] data)
```

Parse a message from an application. Validate the transaction Id and verify that the message hasn't already been received. If the message has multiple buffers, process the incoming buffer. If all of the buffers have been received, place the message on the queue for the application specific thread.
Preconditions Neither session nor data can be null. If cmd is null it is a new command.

---

## handleCableMooseCmd

```
public void handleCableMooseCmd(DA_Session session, byte [] data)
```

---

## makePktHdr

```
public void makePktHdr(byte [] msg, short cmdType, byte sessionId, byte xactId, byte f
```

---

## nakMsg

```
public void nakMsg(byte [] data, short reason)
```

Verify cmd's destination command number, IP address and Port are valid. Send cmd Nak to sender.
Preconditions - Valid cmd parameter.
Postconditions - Not acknowledged message sent to command sender.

---

## nakMsg

```
public void nakMsg(InetAddress addr, int port, short reason)
```

---

## openSession

```
public void openSession()
```

---

## openSessionReply

```
public void openSessionReply()
```

Acknowledge to an openSession message from a client. Returns

---

## recvMsg

```
public DA_Cmd recvMsg()
```

---

## run

```
public void run()
```

Server thread for the network partition. A DA_NetPart will always send and receive messages, regardless of whether it is a server or client partition. This is necessary to support the CableMoose

message protocol.
Semantics - Create a datagram socket and packet. Use a local port assigned by the network stack. Loop forever waiting for incoming messages. Process each message according to whether it is an application message or protocol message.

---

### sendCmdMsg

```
public void sendCmdMsg(AppSession session, DA_Cmd cmd)
```

---

### sendMsg

```
public void sendMsg(AppSession session, DA_Cmd cmd)
```

---

### setSourceVA

```
public void setSourceVA(String newSourceVA)
```

## Association Links

to Class CableMoose_DA.DA_Session

Attribute appSessions
Supplier Cardinality 0..*
Type Composition

CableMoose_DA

# Class DA_Partition

```
CableMoose_DA.DA_Portion
   |
   +--CableMoose_DA.DA_Partition
```

**Direct Known Subclasses:**
DA_NetPart, DA_IPCPart

---

public class **DA_Partition**
extends DA_Portion

Base of the CableMoose communications hierarchy. Inherits from DA_Portion so that DASP or DACP partitions can be assigned to either a communications module or a mutable module that performs application specific processing. Operations defined in this interface will be implemented by deriving classes for realization of the CableMoose proprietary communications protocol.

---

| Fields inherited from class CableMoose_DA.DA_Portion |
|---|
| clientProximityDetector, clientServerParts, peerParts |

## Method Summary

| | |
|---|---|
| void | cmdReply(DA_Cmd cmd) |
| void | destMoveNotify(String oldDestVA, String newDestVA) |
| DA_Cmd | getNextCmd() |

## Method Detail

### cmdReply

```
public void cmdReply(DA_Cmd cmd)
```

---

### destMoveNotify

```
public void destMoveNotify(String oldDestVA, String newDestVA)
```

## getNextCmd

```
public DA_Cmd getNextCmd()
```

CableMoose_DA

# Class DA_Portion

Direct Known Subclasses:
DA_Partition, DASP_Framework, DACP_Framework

---

public class DA_Portion
extends Thread

---

## Field Summary

| | |
|---|---|
| protected ProximityDetector | clientProximityDetector |
| protected java.util.ArrayList | clientServerParts |
| protected java.util.ArrayList | peerParts |

## Field Detail

### clientProximityDetector

```
protected ProximityDetector clientProximityDetector
```

link aggregationByValue
supplierCardinality 1

---

### clientServerParts

```
protected java.util.ArrayList clientServerParts
```

link aggregationByValue
associates
supplierRole Client/Server Partitions
supplierCardinality 0..*

---

### peerParts

```
protected java.util.ArrayList peerParts
```

link aggregationByValue
associates
supplierRole Peer Paritions
supplierCardinality 0..*

## Association Links

to Class CableMoose_DA.ProximityDetector

Attribute clientProximityDetector
Supplier Cardinality 1
Type Composition

to Class CableMoose_DA.DA_Partition

Attribute peerParts
Supplier Role Peer Paritions
Supplier Cardinality 0..*
Type Composition

to Class CableMoose_DA.DA_Partition

Attribute clientServerParts
Supplier Role Client/Server Partitions
Supplier Cardinality 0..*
Type Composition

CableMoose_DA

# Class DA_Session

public class **DA_Session**
extends Object

## Field Summary

| | |
|---|---|
| private byte | CMD_HIST_SIZE |
| HashMap | inboundCmds |
| private byte | lastCmdXactId |
| private String | originatorVA |
| HashMap | outboundCmds |
| private int | sessionId |
| private DatagramPacket | udpPack |
| private DatagramSocket | udpSock |

## Constructor Summary

| |
|---|
| DA_Session(byte newSessionId, InetAddress address, int port) |
| DA_Session(byte newSessionId, String origVA) |

## Method Summary

| | |
|---|---|
| byte | getLastCmdXactId() |
| void | setLastCmdXactId(byte newLastCmdXactId) |
| boolean | validXactId(byte [] data, boolean multiBufs) |

## Field Detail

### CMD_HIST_SIZE

```
private final byte CMD_HIST_SIZE
```

---

### inboundCmds

```
public HashMap inboundCmds
```

link aggregationByValue
associates
supplierCardinality 0..*

---

### lastCmdXactId

```
private byte lastCmdXactId
```

---

### originatorVA

```
private String originatorVA
```

---

### outboundCmds

```
public HashMap outboundCmds
```

link aggregationByValue
associates
supplierCardinality 0..*

---

### sessionId

private int sessionId

---

### udpPack

private DatagramPacket udpPack

---

### udpSock

private DatagramSocket udpSock

## Constructor Detail

### DA_Session

public DA_Session(byte newSessionId, InetAddress address, int port)

---

### DA_Session

public DA_Session(byte newSessionId, String origVA)

## Method Detail

### getLastCmdXactId

public byte getLastCmdXactId()

---

### setLastCmdXactId

public void setLastCmdXactId(byte newLastCmdXactId)

---

### validXactId

public boolean validXactId(byte [] data, boolean multiBufs)

# Association Links

to Class CableMoose_DA.DA_Cmd

Attribute inboundCmds
Supplier Cardinality 0..*

Type Composition

to Class CableMoose_DA.DA_Cmd

Attribute outboundCmds
Supplier Cardinality 0..*
Type Composition

CableMoose_DA

# Class DACP_Framework

CableMoose_DA.DA_Portion
  |
  +--CableMoose_DA.DACP_Framework

---

public class **DACP_Framework**
extends DA_Portion

Base of the OpenCable compliant DACP Framework hierarchy. Contains a fixed number of Peer and Client/Server Partitions as created by the deriving application specific class.

---

| Fields inherited from class CableMoose_DA.DA_Portion |
| --- |
| clientProximityDetector, clientServerParts, peerParts |

## Method Summary

| | |
| --- | --- |
| void | run()<br>Implements a client state machine as defined by the OpenCable middleware specification. |

## Method Detail

### run

```
public void run()
```

Implements a client state machine as defined by the OpenCable middleware specification.

CableMoose_DA

# Class DASP_Framework

```
CableMoose_DA.DA_Portion
   |
   +--CableMoose_DA.DASP_Framework
```

---

public class **DASP_Framework**
extends DA_Portion

Base of the load balancing DA Server portion inheritance hierarchy. Runs as a Thread. Contains a fixed number of Peer and Client/Server Partitions as created by the deriving application specific class.

---

| Fields inherited from class CableMoose_DA.DA_Portion |
| --- |
| clientProximityDetector, clientServerParts, peerParts |

## Method Summary

| | |
| --- | --- |
| void | run()<br>Implements the CableMoose proprietary load balancing server Framework.. |

## Method Detail

### run

```
public void run()
```

Implements the CableMoose proprietary load balancing server Framework..

CableMoose_DA

# Class MooseCmdTypes

public class MooseCmdTypes

## Field Summary

| | |
|---|---|
| static int | APP_CMD_MIN |
| static int | CLOSE_SESSION_CMD |
| static int | CLOSE_SESSION_REPLY_CMD |
| static int | DATA_MSG_REPLY_CMD |
| static int | KILL_CMD_CMD |
| static int | OPEN_SESSION_CMD |
| static int | OPEN_SESSION_REPLY_CMD |
| static int | RESERVED_CMD_MASK |

## Field Detail

### APP_CMD_MIN

```
public static final int APP_CMD_MIN
```

### CLOSE_SESSION_CMD

```
public static final int CLOSE_SESSION_CMD
```

### CLOSE_SESSION_REPLY_CMD

public static final int CLOSE_SESSION_REPLY_CMD

---

## DATA_MSG_REPLY_CMD

public static final int DATA_MSG_REPLY_CMD

---

## KILL_CMD_CMD

public static final int KILL_CMD_CMD

---

## OPEN_SESSION_CMD

public static final int OPEN_SESSION_CMD

---

## OPEN_SESSION_REPLY_CMD

public static final int OPEN_SESSION_REPLY_CMD

---

## RESERVED_CMD_MASK

public static final int RESERVED_CMD_MASK

CableMoose_DA

# Class MooseFault

public class MooseFault

## Field Summary

| | |
|---|---|
| static short | BUF_OUT_OF_ORDER |
| static short | ERR_RETRIES_EXCEEDED |
| static short | INVALID_HDR |
| static short | INVALID_MSG_LEN |
| static short | INVALID_SESSION_ID |
| static short | INVALID_XACT_ID |

## Method Summary

| | |
|---|---|
| static void | log(DA_Cmd cmd, short error) |
| static void | log(DA_Msg msg, short error) |

## Field Detail

### BUF_OUT_OF_ORDER

```
public static short BUF_OUT_OF_ORDER
```

### ERR_RETRIES_EXCEEDED

```
public static short ERR_RETRIES_EXCEEDED
```

### INVALID_HDR

```
public static short INVALID_HDR
```

---

### INVALID_MSG_LEN

```
public static short INVALID_MSG_LEN
```

---

### INVALID_SESSION_ID

```
public static short INVALID_SESSION_ID
```

---

### INVALID_XACT_ID

```
public static short INVALID_XACT_ID
```

## Method Detail

---

### log

```
public static void log(DA_Cmd cmd, short error)
```

---

### log

```
public static void log(DA_Msg msg, short error)
```

CableMoose_DA

# Class ProximityDetector

public class **ProximityDetector**

Stand-alone object that can determine the proximity of one DA Virtual Address to another. It can also determine the VA of the caller, (sourceVA).

## Method Summary

| | |
|---|---|
| boolean | destLocal(String sourceVA, String destVA) |
| String | sourceVA() |

## Method Detail

### destLocal

```
public boolean destLocal(String sourceVA, String destVA)
```

### sourceVA

```
public String sourceVA()
```